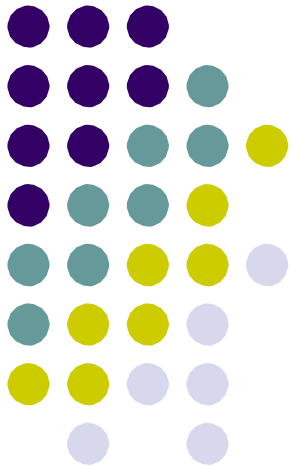
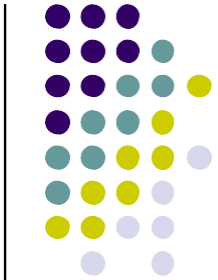
SWEN7302
Secure Software Development

Introduction

Sep 15, 2017

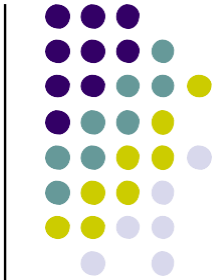


Instructor



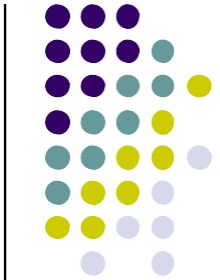
- Ahmad Alsadeh
- Room: 218, Masri Building
- Phone: +970 2 298 2935
- E-mail: asadeh@birzeit.edu
- Office Hours:
  - Check Ritaj
  - By appointments

Course Objectives



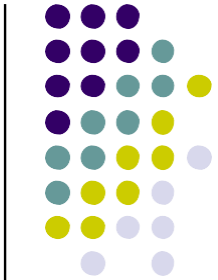
- To learn about how to design/implement secure software
  - Understand and analyze code for vulnerabilities
  - Secure architectures & security assurance
- Understand the principles and practice towards designing secure software
  - Life cycle models/ security engineering principles
  - Usability issues
- To learn about the tools/techniques towards assurance (validation/verification/ testing)
  - Use of tools/techniques to detect coding/design flaws;
  - architectural risk analysis

Course Coverage



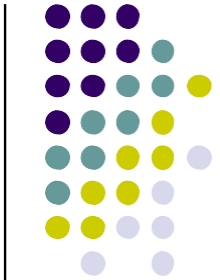
- Threats, attacks, and vulnerabilities that create the need for software security
  - Buffer overflows
  - Input validation
  - Cross-site scripting
- Secure software development & Assurance process
  - Security Engineering/Lifecycle models
    - E.g. Capability Maturity Models and Extensions, Building security In
  - Secure Design/Implementation Principles
    - Systems / software & Formal methods and testing
- Secure Supply Chain environments
- Verification / model checking
- Reverse engineering
- Trusted computing modules/environments

Pre-requisite



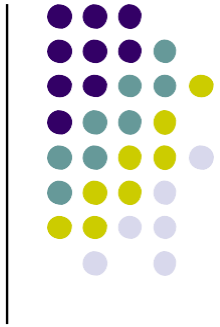
- SWEN6301: Software Construction
- Following courses are preferred but not required:
  - Cryptography and Network Security

Grading (Tentative)



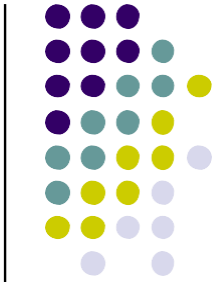
- Homework Exercises 40%
- Research project 25%
- Presentation 5%
- Final Exam 30%

Course Policy



- Attendance is necessary
- Your work MUST be your own
  - No tolerance for cheating/plagiarism
  - Discussing the problem is encouraged
- Homework
  - Penalty for late assignments
  - Ensure clarity in your answers
- Check Ritaj for updates

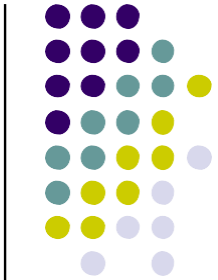
Homework exercises



- Each week, we will cover a different type of code-level vulnerability
  - Some of labs will be taken from the Software security Labs http://www.cis.syr.edu/~wedu/seed/software_security.html

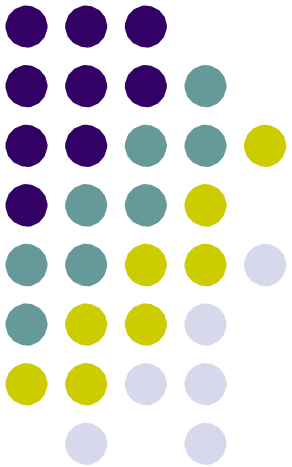
- Each week, we will cover a different type of code-level vulnerability
- Most will link to the Common Weakness Enumeration
  - http://cwe.mitre.org
  - 2011 CWE/SANS Top 25 Most Dangerous Software Errors http://cwe.mitre.org/top25/

Paper writing

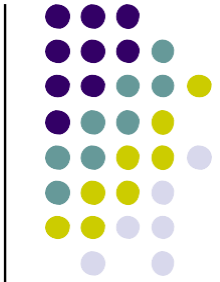


- Security risks of the domain
- Design risks
- Code inspection results
- More details next week

Why Secure Software Development?

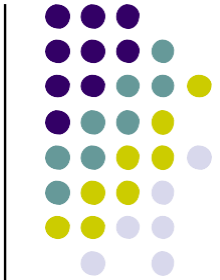


Software/Systems Security



- Renewed ---- interest & importance
  - "*idea of engineering software so that it continues to function correctly under malicious attack*"
  - Existing software is breached with design flaws and implementation bugs
  - "any program, no matter how innocuous it seems, can harbor security holes" [Cheswick & Bellovin, 1994]

Some Common Types of Software Weaknesses:



- Buffer Overflows, Format Strings, Etc.
- Structure and Validity Problems
- Common Special Element Manipulations
- Channel and Path Errors
- Handler Errors
- User Interface Errors
- Pathname Traversal and Equivalence Errors
- Authentication Errors
- Resource Management Errors
- Insufficient Verification of Data
- Code Evaluation and Injection
- Randomness and Predictability

http://cwe.mitre.org/about/index.html

SourceFire (over 25 years)

- 25 years of vulnerabilities (1988 – 2012)
  http://maxedv.com/wp-content/uploads/2011/12/Sourcefire-25-Years-of-Vulnerabilities-Research-Report.pdf

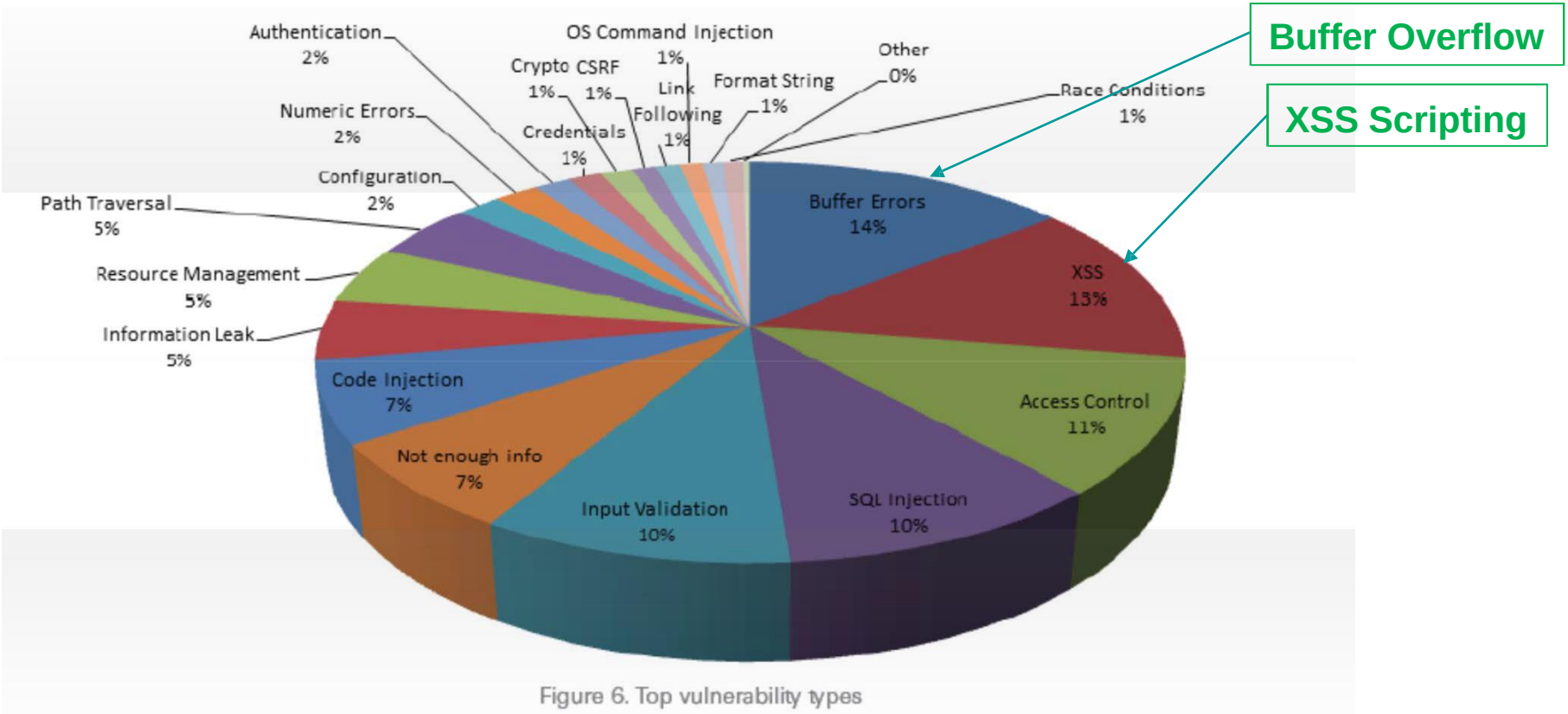
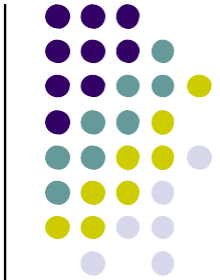


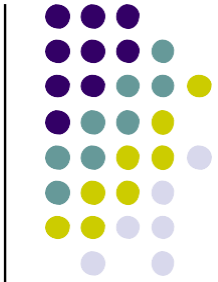
Figure 6. Top vulnerability types

Software security



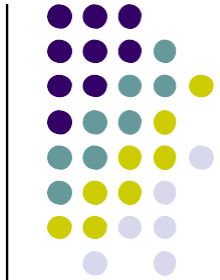
- It is about
  - Understanding software-induced security risks and how to manage them
  - Leveraging software engineering practice,
  - Thinking security early in the software lifecyle
  - Knowing and understanding common problems
  - Designing for security
  - Subjecting all software to exhaustive objective risk analyses and testing
- It is a knowledge intensive field

Security problems in software



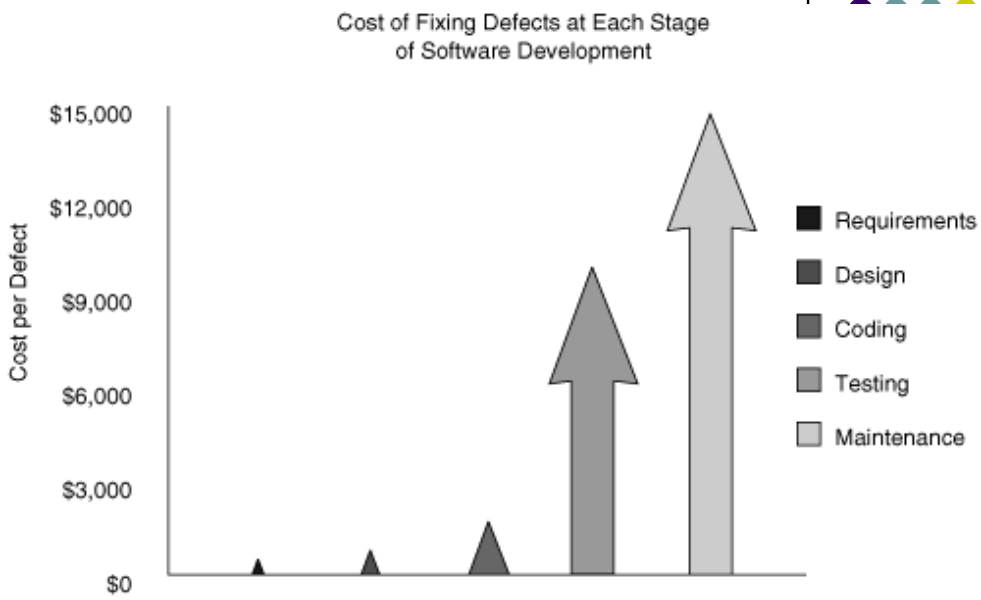
- Defect
  - Implementation and design vulnerabilities
  - Can remain inactive
- Bug
  - An implementation level software problem
- Flaw
  - A problem at a deeper level
- Bugs + Flaws
  - Leads to Risk

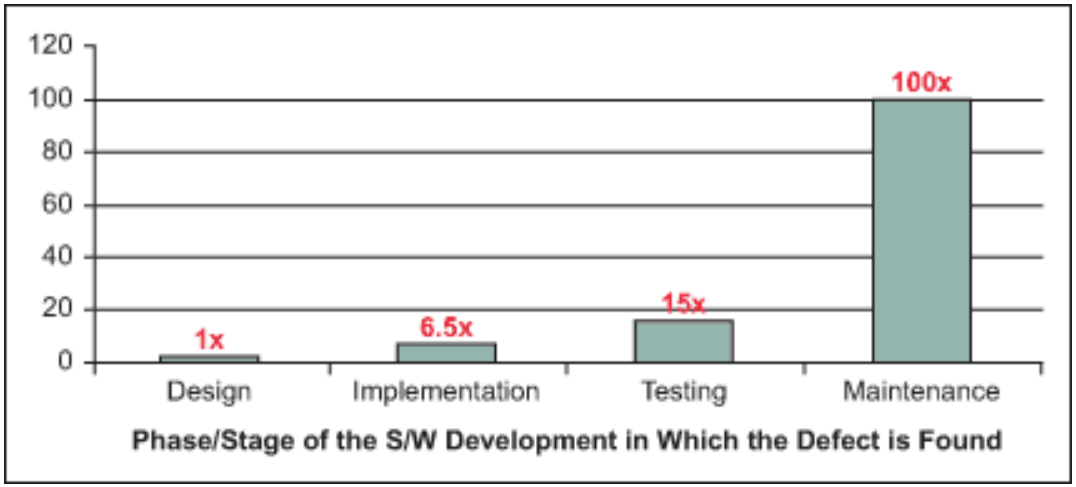
Bugs + Flaws



Bug	Flaw
Buffer overflow: stack smashing Buffer overflow: one-stage attacks Buffer overflow: string format attacks Race conditions: TOCTOU Unsafe environment variables Unsafe system calls (fork(), exec(), system()) Incorrect input validation (black list vs. white list	Method over-riding problems (subclass issues) Compartmentalization problems in design Privileged block protection failure (DoPrivilege()) Error-handling problems (fails open) Type safety confusion error Insecure audit log design Broken or illogical access control (role-based access control [RBAC] over tiers) Signing too much code

Cost of fixing





Relative Costs to Fix Software Defects (Source: IBM Systems Sciences Institute)
https://www.ibm.com/devops/method/experience/deliver/dibbe_edwards_devops_shift_left/

OWASP Top Ten Vulnerabilities (for 2017)

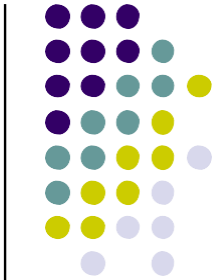
- A1-Injection
- A2-Broken Authentication and Session Management
  - Incorrect implementation (compromise passwords, keys, implementation flaws
- A3-Cross-Site Scripting (XSS)
  - Improper validation
- A4 Broken Access Control (As it was in 2004)
- A5-Security Misconfiguration

OWASP Top Ten Vulnerabilities (for 2017)

- A6-Sensitive Data Exposure
- A7 Insufficient Attack Protection (NEW)
- A8 Cross-Site Request Forgery (CSRF)
  - Forged HTTP requests and compromise of victim's session cookie
  - Victim's browser is forced to generate requests to the vulnerable application
- A9-Using Components with Known Vulnerabilities
  - Components could run with full privileges – vulnerable program could be exploited
  - Components could be libraries or software modules and frameworks
- A10 Underprotected APIs (NEW)
- https://www.owasp.org/index.php/File:OWASP_Top-10_2013.pptx

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2017_Release_Candidate_1

Hence we need …



- Robust and Secure Software Design and Secure Systems Engineering practice
  - Secure development life-cycle/methodologies
  - Secure process models to support large scale team management
  - Fix flaw early in the life-cycle – LOW COST !!
- Secure Design principles & Secure coding practices/standards
- Proper Testing and Verification/Validation
- Effective Tools and Techniques
- Security Engineering education
- Etc..